(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－353960
（P2002－353960A）
(43)公開日　平成14年12月６日(2002.12.6)

(51)Int.Cl.⁷		識別記号	FI		テーマコード(参考)	
Ｈ０４Ｌ	9/32		Ｇ０６Ｆ	9/46	３６０Ｂ	５Ｂ０１７
Ｇ０６Ｆ	1/00			12/14	３２０Ｂ	５Ｂ０３３
	9/318			15/16	６１０Ｚ	５Ｂ０４５
	9/46	３６０	Ｈ０４Ｌ	9/00	６７５Ｄ	５Ｂ０７６
	12/14	３２０			６０１Ｃ	５Ｂ０９８

審査請求　未請求　請求項の数10　ＯＬ　（全 18 頁）　最終頁に続く

(21)出願番号　特願2001－162271(P2001－162271)

(22)出願日　平成13年５月30日(2001.5.30)

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　蒲田　順
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(72)発明者　小谷　誠剛
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

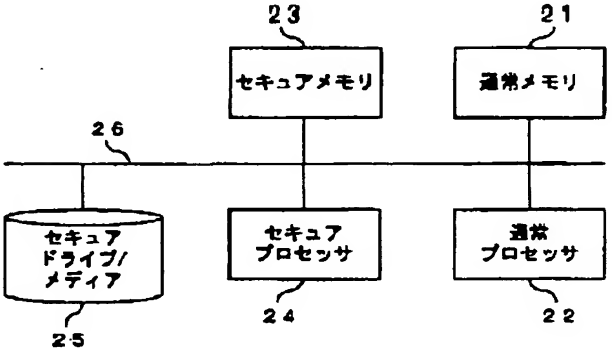
(74)代理人　100074099
弁理士　大菅　義之　（外１名）

最終頁に続く

(54)【発明の名称】　コード実行装置およびコード配布方法

(57)【要約】
【課題】　既存ＯＳを大幅に変更することなく、電子署名と暗号化が施されたコードを効率よく実行することが課題である。
【解決手段】　セキュアプロセッサ２４と通常プロセッサ２２を含むヘテロなマルチプロセッサシステムを構築し、セキュアタスクと非セキュアタスクを各プロセッサへ振り分ける。セキュアタスクの暗号化コードは、セキュアメモリ２３に格納され、セキュアメモリ２３は、認証局の公開鍵で署名を検証して、暗号化コードが有効であることをセキュアプロセッサ２４に通知する。セキュアプロセッサ２４は、セキュアメモリ２３から暗号化命令をフェッチし、復号して実行する。

第１のマルチプロセッサシステムの構成図



1

【特許請求の範囲】

【請求項１】　マルチプロセッサシステムを用いたコード実行装置であって、

セキュアタスクの暗号化コードと、該暗号化コードが正当であることを検証するための検証用情報とを格納するセキュアメモリ手段と、

前記検証用情報により前記暗号化コードが正当であると検証されたとき、該暗号化コードを実行するセキュアプロセッサ手段と、

通常タスクのコードを格納する通常メモリ手段と、

前記通常タスクのコードを実行する通常プロセッサ手段と、

前記セキュアタスクと通常タスクの振り分けを行って、前記暗号化コードを前記セキュアメモリ手段に格納し、前記通常タスクのコードを前記通常メモリ手段に格納する制御手段とを備えることを特徴とするコード実行装置。

【請求項２】　前記セキュアメモリ手段は、物理メモリ割り当ての単位毎に前記暗号化コードを格納し、該単位毎の暗号化コードに対する検証用情報を格納し、該検証用情報を用いて該単位毎の暗号化コードを検証し、前記セキュアプロセッサ手段は、正当であると検証された暗号化コードに含まれる暗号化命令をフェッチし、復号して実行することを特徴とする請求項１記載のコード実行装置。

【請求項３】　前記暗号化コードを固有鍵でさらに暗号化して格納するセキュアドライブ手段をさらに備え、該セキュアドライブ手段と前記セキュアメモリ手段は、相互認証を行ってセッション鍵を共有し、該セキュアドライブ手段は、前記制御手段からの読み出し指示に基づいて、該暗号化コードを該固有鍵で復号し、該セッション鍵で暗号化して、該セキュアメモリ手段に転送することを特徴とする請求項１記載のコード実行装置。

【請求項４】　前記セキュアプロセッサ手段は、前記暗号化コードを用いて、該暗号化コードを実行する論理回路の少なくとも一部を、回路状態として不揮発的に固定することを特徴とする請求項１記載のコード実行装置。

【請求項５】　物理メモリ割り当ての単位毎に暗号化コードを格納する手段と、

該単位毎の暗号化コードが正当であることを検証するための検証用情報を格納する手段と、

該検証用情報を用いて該単位毎の暗号化コードを検証する手段とを備えることを特徴とするメモリ。

【請求項６】　暗号化コードを格納するメモリから、該暗号化コードが正当であることを示す通知を受け取る手段と、

前記通知を受け取ったとき、前記暗号化コードに含まれる暗号化命令をフェッチして復号する手段と、

復号された命令を実行する手段とを備えることを特徴とするプロセッサ。

2

【請求項７】　コンピュータのためのプログラムを記録した記録媒体であって、該プログラムは、

セキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含むマルチプロセッサシステムにおいて、該セキュアタスクと通常タスクの振り分けを行い、

前記セキュアタスクの暗号化コードと、該暗号化コードが正当であることを検証するための検証用情報とを、セキュアメモリに格納し、

前記検証用情報により前記暗号化コードが正当であると検証されたとき、該暗号化コードをセキュアプロセッサに実行させる処理を前記コンピュータに実行させることを特徴とするコンピュータ読み取り可能な記録媒体。

【請求項８】　セキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含むマルチプロセッサシステムにおいて、該セキュアタスクと通常タスクの振り分けを行い、

前記セキュアタスクの暗号化コードと、該暗号化コードが正当であることを検証するための検証用情報とを、セキュアメモリに格納し、

前記検証用情報により前記暗号化コードが正当であると検証されたとき、該暗号化コードをセキュアプロセッサに実行させる処理をコンピュータに実行させるためのプログラム。

【請求項９】　コード作成者が、コード認証機関に実行可能なコードを提供し、

前記コード認証機関が、前記コードが正当であることを検証するための検証用情報を該コードに付加して、マルチプロセッサシステムのユーザに配布し、

前記マルチプロセッサシステムは、前記コードを用いてセキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含み、該セキュアタスクと通常タスクの振り分けを行い、前記検証用情報を用いて該コードが正当であることを検証し、該コードを実行することを特徴とするコード配布方法。

【請求項１０】　コード作成者が、コード認証機関に実行可能なコードを提供して、手数料を支払い、

前記コード認証機関が、前記コードが正当であることを検証するための検証用情報を該コードに付加し、

前記コード作成者が、前記コードをマルチプロセッサシステムのユーザに配布して、該ユーザが支払う対価を受領し、

前記マルチプロセッサシステムは、前記コードを用いてセキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含み、該セキュアタスクと通常タスクの振り分けを行い、前記検証用情報を用いて該コードが正当であることを検証し、該コードを実行することを特徴とするコード配布方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、コード化されたプログラムのような、実行可能なコードを実行する装置およびそのようなコードを配布する方法に関する。

【０００２】

【従来の技術】電子署名と暗号化が施された実行可能なコード（以下では、単にコードと記す）を、署名を検証した上で復号しながら実行することで、認証されたコードのみが安全に動作する環境が実現できる。実行可能なコードには、コード化されたプログラムの一部または全体が含まれる。このような動作環境の実現方法として、署名検証機能と復号機能を持つプロセッサ（セキュアプロセッサ）をメモリマップドの入出力装置（Ｉ／Ｏ装置）として見せかける方法が考えられる。この方法では、暗号化コードをデータとしてそのＩ／Ｏ装置に渡し、実行させて、実行結果を取得する。

【０００３】

【発明が解決しようとする課題】しかしながら、上述した従来のコード実行方法には、次のような問題がある。このコード実行方法では、Ｉ／Ｏ装置のメモリ量の制限からサイズの大きなコード全体を一度に渡すことができない。また、一旦コードを渡して実行を開始すると、終わるまで他の制御を行うことができないため、署名検証や復号等のセキュア処理をマルチタスクで実行することができない。このため、セキュア処理を伴う複数のタスクを効率よく実行することができない。

【０００４】このうち、後者の問題は、Ｉ／Ｏ装置をマルチタスク処理が可能なように設計し、そのＩ／Ｏ装置専用のタスク管理モジュールをオペレーティングシステム（ＯＳ）内に用意することで解決できる。しかし、ＯＳが、Ｉ／Ｏ装置（セキュアプロセッサ）専用のタスク管理モジュールを、通常プロセッサ用のタスク管理モジュールと重複して持つことになり、効率上好ましくない。

【０００５】さらに、セキュリティ上の観点から、ＯＳそのものがセキュアプロセッサ上で動作することが望ましいが、既存ＯＳをいきなりセキュアプロセッサ用に書き換えることは簡単ではないという問題もある。

【０００６】本発明の課題は、既存ＯＳを大幅に変更することなく、電子署名と暗号化が施されたコードを効率よく実行する装置と、そのような装置に対してコードを配布する方法を提供することである。

【０００７】

【課題を解決するための手段】図１は、本発明のコード実行装置の原理図である。図１のコード実行装置は、マルチプロセッサシステムを用いて実現され、セキュアメモリ手段１１、セキュアプロセッサ手段１２、通常メモリ手段１３、通常プロセッサ手段１４、および制御手段１５を備える。

【０００８】セキュアメモリ手段１１は、セキュアタスクの暗号化コードと、その暗号化コードが正当であることを検証するための検証用情報とを格納する。セキュアプロセッサ手段１２は、検証用情報により暗号化コードが正当であると検証されたとき、その暗号化コードを実行する。通常メモリ手段１３は、通常タスクのコードを格納し、通常プロセッサ手段１４は、通常タスクのコードを実行する。

【０００９】制御手段１５は、セキュアタスクと通常タスクの振り分けを行って、暗号化コードをセキュアメモリ手段１１に格納し、通常タスクのコードを通常メモリ手段１３に格納する。

【００１０】検証用情報としては、例えば、電子署名、パリティコード、ＣＲＣ（Cyclic Redundancy Check）ビットが用いられる。また、制御手段１５は、例えば、マルチプロセッサシステムのＯＳに対応する。

【００１１】セキュアタスクを実行する場合、制御手段１５は、その暗号化コードを検証用情報とともにセキュアメモリ手段１１に格納する。そして、検証用情報を用いて暗号化コードが正当であると検証されると、セキュアプロセッサ手段１２が、その暗号化コードを実行する。

【００１２】また、通常タスクを実行する場合、制御手段１５は、そのコードを通常メモリ手段１３に格納する。そして、通常プロセッサ手段１４は、そのコードを実行する。

【００１３】このように、マルチプロセッサシステム内でセキュアタスクと通常タスクの共存を許し、制御手段１５がそれらのタスクをセキュアプロセッサ手段１２と通常プロセッサ手段１４に振り分けることで、タスク管理が簡単になる。したがって、ＯＳを大幅に変更することなく、セキュアタスクのコードを効率よく実行することができる。

【００１４】また、このようなマルチプロセッサシステムに対してコードを配布する方法としては、例えば、以下の２通りの方法がある。

（１）コード作成者が、コード認証機関に実行可能なコードを提供し、コード認証機関が、コードが正当であることを検証するための検証用情報をコードに付加して、マルチプロセッサシステムのユーザに配布する。

（２）コード作成者が、コード認証機関に実行可能なコードを提供して、手数料を支払い、コード認証機関が、検証用情報をコードに付加する。その後、コード作成者が、コードをマルチプロセッサシステムのユーザに配布して、ユーザが支払う対価を受領する。

【００１５】

【発明の実施の形態】以下、図面を参照しながら、本発明の実施の形態を詳細に説明する。本実施形態では、マルチプロセッサシステム中の１つのプロセッサをセキュアプロセッサに置き換え、ヘテロなマルチプロセッサシステムを構築する。そして、セキュアなタスクと非セキュアなタスクの各プロセッサへの割り当てを、ＯＳが制

御する。このように、セキュアプロセッサを含むヘテロなマルチプロセッサシステムを構築することで、ＯＳが２種類のタスク管理モジュールを重複して持つ必要がなくなる。

【００１６】また、セキュアタスクと非セキュアタスクの共存を許すことで、可能な部分からセキュアタスク化していき、最終的にＯＳの全タスクをセキュア化するというマイグレーションが可能となる。このようなマイグレーションは、ＯＳが細かなタスクの集合体として実現されているような場合に限られるが、これにより、既存ＯＳをいきなりセキュアプロセッサ用に書き換える必要がなくなる。

【００１７】しかし、このようなシステムにおいては、セキュアプロセッサが暗号化コードを１命令ずつ取り出して（フェッチして）実行するため、コード全体をセキュアプロセッサにまとめて渡すことはできない。したがって、暗号化コード全体に署名をつけたのでは、署名の検証ができないという問題が発生する。そこで、本実施形態では、メモリ割り当ての最小単位であるページ（例えば、４Ｋバイト等）単位に署名を付与した形で暗号化コードを作成し、メモリ割り当て時にメモリ自身が署名を検証する構成を採用する。

【００１８】さらに、このシステムに対して配布すべきコードを収集し、収集したコードに署名をつけて配布する機関を設ける。これにより、コード作成者は、コードを広範囲に配布することが可能となり、ユーザは、安心してコードを利用することが可能となる。

【００１９】図２は、このようなマルチプロセッサシステムの構成図である。図２のシステムは、通常メモリ２１、通常プロセッサ２２、セキュアメモリ２３、セキュアプロセッサ２４、およびセキュアドライブ／メディア２５を含む。これらの装置はシステムバス２６により互いに接続されているが、通常プロセッサ２２は、セキュアメモリ２３から命令をフェッチすることはなく、セキュアプロセッサ２４は、通常メモリ２１から命令をフェッチすることはない。

【００２０】通常プロセッサ２２は、通常メモリ２１を用いて通常のタスク（非セキュアタスク）の通常のコードを実行し、セキュアプロセッサ２４は、セキュアメモリ２３を用いてセキュアタスクの暗号化コードを実行する。セキュアドライブ／メディア２５は、セキュアタスク用の暗号化コードを格納する格納装置である。図２では、通常プロセッサ２２とセキュアプロセッサ２４が１つずつ設けられているが、各プロセッサを複数設けてもよい。

【００２１】図３は、セキュアメモリ２３とセキュアプロセッサ２４の構成例を示している。図３のセキュアメモリ２３は、認証局公開鍵３１、署名検証部３２、署名保持部３３、ページ３４を有する。ページ３４は、物理メモリ（実メモリ）割り当ての最小単位であり、例えば、４Ｋバイトの容量を持つ。また、署名保持部３３は、ページ毎に、ページに対する署名データを格納する領域を持つ。署名検証部３２は、例えば、ハードウェアまたはＭＰＵ（Micro Processing Unit ）を用いて実装され、ページ単位で署名を用いて暗号化コードを検証する。

【００２２】署名としては、例えば、認証局（Certificate Authority ，ＣＡ）の秘密鍵によって作成され、セキュアメモリ２３にあらかじめ格納されたＣＡの公開鍵によって検証可能なＸ．５０９証明書が用いられる。

【００２３】セキュアプロセッサ２４は、復号鍵設定部４１、復号鍵保持部４２、復号部４３、およびプロセッサ４４を有する。このうち、復号鍵設定部４１、復号鍵保持部４２、および復号部４３は、命令を実行するプロセッサ４４の命令入力部の前段に設けられ、復号部４３は、例えば、ハードウェアまたはＭＰＵを用いて実装される。

【００２４】署名を含む暗号化コードは、セキュアドライブ／メディア２５より読み出され、署名と暗号化コードに分離された後、それぞれ署名保持部３３とページ３４に格納される。

【００２５】これらの格納が終了すると、署名検証部３２は、署名を作成した認証局の公開鍵で署名を検証し、問題がなければ、署名に含まれる暗号化コードのハッシュ値を、ページ３４上の暗号化コードから再計算したハッシュ値と比較する。

【００２６】これらのハッシュ値が一致し、暗号化コードが改竄されていないことを確認すると、ページ３４上の暗号化コードが有効である（正当なコードである）ことをセキュアプロセッサ２４に通知する。また、ハッシュ値が一致しなければ、ページ３４上の暗号化コードが無効である（正当なコードでない）ことをセキュアプロセッサ２４に通知する。

【００２７】セキュアプロセッサ２４の復号部４３は、有効の通知を受け取ると、ページ３４上のメモリアドレスから必要な暗号化命令をフェッチし、復号鍵保持部４２の復号鍵を用いて暗号化命令を順次復号する。そして、プロセッサ４４は、復号された命令を順次実行する。復号に必要な復号鍵は、復号鍵設定部４１により、あらかじめ復号鍵保持部４２に設定されている。

【００２８】さらに、セキュアプロセッサ２４に複数の復号鍵を格納しておき、どの復号鍵を使用して復号するかを外部より指定可能とすることもできる。図４は、このようなセキュアプロセッサ２４の構成図である。

【００２９】図４のセキュアプロセッサ２４は、複数の復号鍵保持部４２を持つ点と、これらの保持部４２に格納された複数の復号鍵のうちのどれを使用するかを指示する復号鍵指示部４５が追加されている点が、図３のセキュアプロセッサ２４とは異なる。復号鍵指示部４５は、例えば、ハードウェアまたはＭＰＵを用いて実装さ

7

れる。どの復号鍵を使用するかは、例えば、現在実行中のセキュアタスクに応じて、ＯＳが復号鍵指示部４５に指示する。

【００３０】図５は、図２のマルチプロセッサシステムの動作を制御するセキュアＯＳの構成図である。図５のセキュアＯＳ５１は、セキュアプロセッサ２４または／および通常プロセッサ２２上で動作し、セキュアタスク管理５２、セキュアメモリ管理５３、およびセキュアファイルシステム５４を含む。

【００３１】セキュアタスク管理５２とセキュアメモリ管理５３は、セキュアタスクと非セキュアタスクの振り分けを行う。これにより、セキュアタスクの暗号化コードはセキュアメモリ２３に格納され、非セキュアタスクのコードは通常メモリ２１に格納される。また、セキュアタスク管理５２は、セキュアタスクと非セキュアタスクの両方のマルチタスク処理を制御する。以下、対象とするタスクがセキュアタスクの場合の動作を説明する。

【００３２】セキュアタスク管理５２は、複数のタスクのコンテキストを管理し、コンテキスト切り替え時に、セキュアプロセッサ２４のプログラムカウンタの変更等の通常処理のほか、セキュアプロセッサ２４内に保持されている復号鍵のうちどれを使用するかを指示する。

【００３３】セキュアメモリ管理５３は、必要に応じて、セキュアタスクへのセキュアメモリ２３の割り当てを行う。なお、セキュアメモリ２３からセキュアプロセッサ２４への暗号化命令の転送は、ＣＰＵ（Central Processing Unit ）のフェッチ動作であるので、セキュアメモリ管理５３を経由しない。

【００３４】セキュアファイルシステム３４は、セキュアドライブ／メディア２５に格納された暗号化コードのファイルを管理する。そして、セキュアメモリ管理５３からの要求に応じて、セキュアドライブ／メディア２５から暗号化コードを読み出し、セキュアメモリ管理５３に渡す。

【００３５】次に、図６から図９までを参照しながら、セキュアタスク管理５２、セキュアメモリ管理５３、およびセキュアファイルシステム５４の処理をより詳細に説明する。

【００３６】図６は、セキュアタスク管理５２の処理のフローチャートである。図６の処理は、セキュアプロセッサ２４上で現在実行中のセキュアタスクのタイムスライスが切れ、タイマ割り込みが発生して、セキュアタスク管理５２に制御が移ったときに開始される。

【００３７】セキュアタスク管理５２は、まず、スケジューリングアルゴリズムに従って、次に実行するセキュアタスクＡを決定し（ステップＳ１）、セキュアタスクＡのコンテキストを復元する（ステップＳ２）。このとき、セキュアプロセッサ２４のプログラムカウンタおよびスタックポインタの復元や、セキュアプロセッサ２４とセキュアメモリ２３の間に存在するＭＭＵ（Memory Management Unit）内のＴＬＢ（Translation Look aside Buffer︶）の復元等が行われる。

8

【００３８】次に、プログラム復号鍵としてセキュアタスクＡ用のものを使用することを、セキュアプロセッサ２４に指示する（ステップＳ３）。そして、セキュアタスクＡのタイムスライス（例えば、１００ｍｓ）をタイマに設定し（ステップＳ４）、セキュアプロセッサの動作を再開する（ステップＳ５）。

【００３９】図７は、セキュアメモリ管理５３の処理のフローチャートである。図７の処理は、セキュアタスク実行中にページフォルトが起き、割り込みが発生して、セキュアメモリ管理５３に制御が移ったときに開始される。

【００４０】セキュアメモリ管理５３は、まず、セキュアメモリ２３内に未使用の実メモリ領域があるか否かをチェックし（ステップＳ１１）、実メモリがあれば、それを１ページ割り当てる（ステップＳ１３）。未使用の実メモリがない場合は、実メモリ解放処理のサブルーチンを呼び出して（ステップＳ１２）、空きを作った後に実メモリを割り当てる。

【００４１】次に、割り当てた実メモリのアドレスと仮想アドレスの対応表を作成し、ＭＭＵ内のＴＬＢに格納する（ステップＳ１４）。そして、割り当てた実メモリに配置するコードをセキュアファイルシステム５４に要求し、受け取ったコードを実メモリに配置して（ステップＳ１５）、セキュアプロセッサ２４の動作を再開する（ステップＳ１６）。

【００４２】図８は、図７のステップＳ１２において呼び出されたサブルーチンが行う実メモリ解放処理のフローチャートである。サブルーチンは、まず、実メモリ解放アルゴリズムに従って、ページアウトの対象となる実メモリを決定する（ステップＳ２１）。次に、対象となった実メモリ上のコードを、セキュアドライブ／メディア２５にページアウトする（書き出す）（ステップＳ２２）。そして、呼び出し元に復帰する（ステップＳ２３）。

【００４３】図９は、セキュアファイルシステム５４の処理のフローチャートである。図９の処理は、図７のステップＳ１５においてセキュアメモリ管理５３からコードの要求を受けたときに開始される。

【００４４】セキュアファイルシステム５４は、まず、対象となるプログラムの先頭からのオフセットを受け取り（ステップＳ３１）、セキュアドライブ／メディア２５内で指定位置までシークする（ステップＳ３２）。そして、指定位置より１ページ分のコードを読み出し、セキュアメモリ管理５３に渡す（ステップＳ３３）。

【００４５】ところで、セキュアメモリ２３とセキュアプロセッサ２４が相互認証を行ってセッション鍵を共有することで、暗号化コードをより安全にやり取りすることも可能である。この場合、セキュアメモリ２３は、暗

号化コードをセッション鍵でさらに暗号化した上でセキュアプロセッサ２４に転送する。

【００４６】図１０は、このようなセキュアメモリ２３とセキュアプロセッサ２４の構成図である。図１０のセキュアメモリ２３は、図３の構成に加えて、さらに相互認証／セッション鍵共有部６１および暗号部６２を有し、セキュアプロセッサ２４は、図４の構成に加えて、さらに相互認証／セッション鍵共有部７１および復号部７２を有する。

【００４７】まず、相互認証／セッション鍵共有部６１と相互認証／セッション鍵共有部７１は、相互に信頼できる相手であることを認証した上で、セッション鍵を生成／共有する。相互認証の方法は、公開鍵を用いた証明書ベースの方法でもかまわないし、共通鍵を用いた方法でもかまわない。また、セッション鍵は、例えば、乱数を用いて生成される。

【００４８】その後、セキュアメモリ２３の暗号部６２は、ページ３４上の暗号化命令をセッション鍵でさらに暗号化して、セキュアプロセッサ２４に転送する。セキュアプロセッサ２４の復号部７２は、受け取った暗号化命令をセッション鍵で復号した後、復号部４３に渡す。その後、図４に示したように、暗号化命令が対応する復号鍵で復号されて実行される。

【００４９】また、同様にして、セキュアドライブ／メディア２５とセキュアメモリ２３が相互認証を行ってセッション鍵を共有することで、暗号化コードをより安全にやり取りすることも可能である。

【００５０】図１１は、このようなセキュアドライブ／メディア２５とセキュアメモリ２３の構成図である。図１１のセキュアドライブ／メディア２５は、格納媒体８１、格納装置固有鍵８２、相互認証／セッション鍵共有部８３、復号部８４、および暗号部８５を有し、セキュアメモリ２３は、図１０の構成に加えて、さらに復号部６３を有する。

【００５１】セキュアドライブ／メディア２５は、暗号化コードを格納装置固有鍵８２または格納媒体固有鍵８６でさらに暗号化して、格納媒体８１に格納する。格納媒体８１としては、磁気ディスク、光ディスク、光磁気ディスク、磁気テープ等が用いられる。また、格納装置固有鍵８２は、セキュアドライブ／メディア２５固有の鍵であり、格納媒体固有鍵８６は、格納媒体８１固有の鍵である。

【００５２】相互認証／セッション鍵共有部８３と相互認証／セッション鍵共有部６１は、図１０の場合と同様にして、相互に信頼できる相手であることを認証した上で、セッション鍵を生成／共有する。

【００５３】セキュアドライブ／メディア２５の復号部８４は、格納媒体８１上に格納された暗号化コード８７を、格納装置固有鍵８２または格納媒体固有鍵８６で復号して、暗号部８５に渡す。暗号部８５は、相互認証／セッション鍵共有部８３が保持しているセッション鍵により暗号化コードをさらに暗号化して、セキュアメモリ２３に転送する。セキュアメモリ２３の復号部６３は、受け取った暗号化コードをセッション鍵で復号して、元の暗号化コードの形に戻した上で、ページ３４に格納する。

【００５４】このとき、図５のセキュアファイルシステム５４は、セキュアドライブ／メディア２５とセキュアメモリ２３の間のセッション鍵の共有を仲介する。その後、セキュアファイルシステム５４は、このセッション鍵によって暗号化された暗号化コードを、格納媒体８１上の論理フォーマットに従ってセキュアドライブ／メディア２５から読み出し、セキュアメモリ２３に転送する。

【００５５】ここで、１つのセキュアタスクが実行される場合の処理の流れを説明する。この場合、セキュアタスクには現在セキュアメモリが１ページだけ割り当てられており、プログラムカウンタはそのページ上の暗号化コードの最後の暗号化命令を指しているものとする。また、各エンティティ（セキュアメモリ２３、セキュアプロセッサ２４、セキュアドライブ／メディア２５）間の相互認証処理、セッション鍵の共有処理、セッション鍵による暗号化／復号処理については、説明を省略する。

（１）セキュアプロセッサ２４は、セキュアメモリ２３より暗号化コードをフェッチし、復号した後に実行する。

（２）セキュアプロセッサ２４は、プログラムカウンタをインクリメントし、次の命令のフェッチ動作を実行する。

（３）実メモリが未割り当てであるため、セキュアメモリ２３は、ページフォルト例外をセキュアタスク管理５２に対して発生する。

（４）セキュアタスク管理５２は、実行中のセキュアタスクをスリープ状態に設定した後、セキュアメモリ管理５３に新たな実メモリの割り当てを依頼する。

（５）セキュアメモリ管理５３は、新たな実メモリ１ページをセキュアタスクに割り当てる。

（６）セキュアタスク管理５２は、暗号化コードの続きを読み出すように、セキュアファイルシステム５４に依頼する。

（７）セキュアファイルシステム５４は、暗号化コードの続きをセキュアドライブ／メディア２５より読み出し、新たに割り当てられた実メモリに格納する。

（８）セキュアタスク管理５２は、スリープ状態のセキュアタスクを実行状態に設定する。

（９）セキュアプロセッサ２４は、新たに割り当てられたページ上の次の命令をフェッチして実行する。

【００５６】次に、２つのセキュアタスクＡ、Ｂが実行される場合の処理の流れを説明する。この場合、セキュアタスクＡおよびＢともに、十分なセキュアメモリが割

り当てられており、ページフォルトは起きないものとする。

（１）セキュアプロセッサ２４は、セキュアタスクＡの暗号化命令をフェッチして実行する。

（２）セキュアタスク管理５２は、タイムスライスが切れ、タイマ割り込みが発生したため、セキュアタスクＡをスリープ状態に設定する。

（３）セキュアタスク管理５２は、スケジューリングアルゴリズムに従って、次に動作させるタスクをセキュアタスクＢに決定し、セキュアタスクＢを動作状態に設定する。

（４）セキュアタスク管理５２は、セキュアタスクＢの復号に必要な鍵をセキュアプロセッサ２４に対して指示する。

（５）セキュアタスク管理５２は、プログラムカウンタ、スタックポインタ、ＴＬＢのアドレス対応表等をセキュアタスクＢ用に設定する。

（６）セキュアプロセッサ２４は、セキュアタスクＢの暗号化命令をフェッチして実行する。

【００５７】以上の説明は、セキュアＯＳがセキュアプロセッサ２４上で動作するものと考えると容易に理解できる。しかし、セキュアプロセッサ２４の実行を一時停止する機能や、セキュアプロセッサ２４のプログラムカウンタの変更等のようなコンテキストを切り替える機能がセキュアプロセッサ２４に用意されていれば、セキュアＯＳ自身は通常プロセッサ２２上で動作させることも可能である。

【００５８】図２のマルチプロセッサシステムでは、セキュアメモリ２３と通常メモリ２１が別々に設けられているが、セキュアメモリ２３と通常メモリ２１の一部または全部がオーバーラップした形態も考えられる。

【００５９】図１２および図１３は、このようなマルチプロセッサシステムの構成例を示している。ただし、ここではセキュアドライブ／メディア２５は省略されている。図１２において、セキュアプロセッサ２４と通常プロセッサ２２は同一のシステムバス９２（データバス、アドレスバス）を介して、セキュアメモリ９１に接続されている。この場合、セキュアメモリ９１は、図２のセキュアメモリ２３と通常メモリ２１の機能を兼ね備えている。

【００６０】また、図１３において、セキュアプロセッサ２４は、システムバス９４を介して固有のセキュアメモリ２３に接続されており、システムバス９５を介して共有メモリ９３に接続されている。また、通常プロセッサ２２は、システムバス９６を介して固有の通常メモリ２１に接続されており、システムバス９５を介して共有メモリ９３に接続されている。共有メモリ９３は、セキュアプロセッサ２４と通常プロセッサ２２に共通のメモリであり、セキュアメモリ２３または／および通常メモリ２１の機能を備えている。

【００６１】図１２の構成は、システムバスおよびメモリが１つずつしかないため、図１３の構成よりコストが低いという利点がある。しかし、セキュアプロセッサ２４と通常プロセッサ２２の両方がセキュアメモリ９１にアクセス可能なため、図１３の構成よりセキュリティレベルが低くなる。逆に、図１３の構成は、図１２の構成よりコストが増加するが、セキュリティレベルも向上する。

【００６２】以上の実施形態では、セキュアプロセッサ２４がコードをフェッチして実行しているが、コードの一部または全部を用いて、暗号化命令をフェッチし復号した上で実行する論理回路を自動的に生成することも考えられる。この場合、汎用論理回路を特定の回路状態に固定する装置がシステム内に設けられる。

【００６３】セキュアメモリ２３が正当なコードであることを検証した後、セキュアプロセッサ２４は、そのコードを用いて論理回路の一部または全部を、回路状態として不揮発的に固定する。このとき、前の回路状態を消去して、新しく上書きする。

【００６４】図１４は、このような回路生成処理のフローチャートである。セキュアプロセッサ２４は、まず、暗号化命令をフェッチして復号し（ステップＳ４１）、コードを演算処理回路構成情報に翻訳する（ステップＳ４２）。次に、回路構成情報を配線情報に翻訳し（ステップＳ４３）、配線情報を不揮発的に焼き付ける（ステップＳ４４）。配線情報の焼き付け方法としては、例えば、以下の２通りが考えられる。

（１）図１５に示すように、複数の基本回路を配列（アレイ）状に並べ、配線情報に基づいて回路間を不揮発的に接続し、演算器を構成する。

（２）図１６に示すように、構成済みの基本演算器を多種用意しておき、配線情報に基づいて必要な演算器間を不揮発的に接続する。

【００６５】このように、処理部分をハードウェア化することで処理速度が向上する。また、ハードウェアとソフトウェア処理を併用すれば、暗号化命令を階層化して、セキュリティレベルを向上させることもできる。例えば、特に重要な部分の命令は、厳重な認証ステップを経てハードウェア化しておき、それ以外の命令は、ユーザの利便性を図るため、軽い認証で毎回ソフトウェア処理する。

【００６６】以上の実施形態では、署名を用いてコードが正当か否かが検証されているが、コードが正当であることを検証するための情報（検証用情報）としては、他の任意の情報を用いることができる。例えば、パリティコード、ＣＲＣ（Cyclic Redundancy Check ）ビット等を付加しておくことで、コードが壊れているか否かを検証することができる。そこで、以下では、署名の代わりに検証用情報という言葉を用い、この情報をコードに付加する機関をコード認証機関と呼ぶことにする。

【００６７】次に、図１７から図２２までを参照しながら、検証用情報が付加されたコードの配布方法について説明する。図１７は、ユーザへのコード配布方法を示している。図１７において、コード作成者１０１は、コード認証機関１０２にコードを提供する（Ｐ１）。コード認証機関１０２は、受け取ったコードの正当性を確認した上で、検証用情報を付加し、認証済みコードをコード利用者１０３（ユーザ）に提供する（Ｐ２）。コード利用者１０３は、例えば、上述したマルチプロセッサシステムを保有しており、受け取ったコードに付加された検証用情報を用いてコードの正当性を確認した後に、そのコードを利用する。

【００６８】このとき、コード認証機関１０２は、コード作成者１０１に対して対価を提示してコードを収集し、収集に際して対価を支払う。そして、コード利用者１０３に対してコードの対価を提示し、検証用情報を付加した後、コード利用者１０３に対してコードを提供すると同時に対価を徴収する。

【００６９】図１８は、このような対価の支払いを示している。図１８において、コード作成者１０１は、コード認証機関１０２にコードを提供し（Ｐ１１）、その対価をコード認証機関１０２から受け取る（Ｐ１２）。コード認証機関１０２は、コード利用者１０３に認証済みコードを提供し（Ｐ１３）、コード利用者１０３は、それに対する対価をコード認証機関１０２に支払う（Ｐ１４）。

【００７０】コード利用者１０３およびコード認証機関１０２が支払う対価は、コード提供時に一度に課金されてもかまわないし、コードの利用／提供状況に応じて従量課金されてもかまわない。後者の場合、例えば、コード利用者１０３が受領したコード数に応じて課金が行われる。

【００７１】また、コード作成者１０１がコード認証機関１０２に手数料を支払うことで、コードに検証用情報を付加してもらい、コード利用者１０３が支払う対価を受領することも可能である。

【００７２】図１９は、このようなコード配布方法を示している。図１９において、コード作成者１０１は、コード認証機関１０２にコードを提供し（Ｐ２１）、検証用情報を付加してもらうのに必要な手数料を支払って（Ｐ２２）、認証済みコードを取得する（Ｐ２３）。次に、認証済みコードをコード利用者１０３に提供して（Ｐ２４）、その対価を受け取る（Ｐ２５）。

【００７３】コード利用者１０３が支払う対価は、コード提供時に一度に課金されてもかまわないし、コードの利用／提供状況に応じて従量課金されてもかまわない。また、コード作成者１０１が支払う手数料も、同様に、一括課金でもかまわないし、従量課金でもかまわない。

【００７４】また、コード作成者１０１の代わりに、コード認証機関１０２がコードを配布してもよい。この場合、コード認証機関１０２が、認証済みコードをコード利用者１０３に提供して対価を徴収し、徴収した対価をコード作成者１０１に支払う。

【００７５】また、図１７のコード配布方法において、コード認証機関１０２は、受け取ったコードを２つ以上の部分に分割し、最初に一部を配布し、その後、コード利用者１０３の要求に応じて、残りの部分を配布することも可能である。この場合、最初の配布は、例えば、以下のいずれかの方法で行われる。

（１）複数のユーザにコードを放送する。

（２）各ユーザに、ネットワーク上からコードを自由にダウンロードさせる。

（３）コードを可搬記録媒体に収納して、その記録媒体をユーザに配布する。

【００７６】図２０は、このようなコード配布方法を示している。図２０において、コード作成者１０１は、コード認証機関１０２にコードを提供する（Ｐ３１）。コード認証機関１０２は、コードの正当性を確認した後、検証用情報を付加して、コード利用者１０３に認証済みコードの一部を提供する（Ｐ３２）。コード利用者１０３は、提供された一部のコードが正当であることを検証用情報を用いて確認した後、そのコードを利用する。さらに、必要であれば、残りの認証済みコードをコード認証機関１０２より入手して利用する（Ｐ３３）。

【００７７】最初に提供されるコードは、例えば、印刷機能に制限を設けた年賀状作成ソフトや、最初の画面データのみを収録したゲームソフト等である。残りのコードは、例えば、すべての機能制限を取り除いた年賀状作成ソフトや、第２画面以降を収録したゲームソフト等である。

【００７８】このとき、コード認証機関１０２は、コード作成者１０１に対して対価を提示してコードを収集し、収集に際して対価を支払う。次に、コード利用者１０３に対して残しておいた部分のコードの対価を提示し、検証用情報を付加した後、コードを提供すると同時に対価を徴収する。

【００７９】図２１は、このような対価の支払いを示している。図２１において、Ｐ４１およびＰ４２の処理は、図１８のＰ１１およびＰ１２の処理と同様である。次に、コード認証機関１０２は、認証済みコードの一部を、例えば、雑誌付録のＣＤ－ＲＯＭ（compact disk read only memory）やインターネットを通じて、無償で配布する（Ｐ４３）。これを入手して利用したコード利用者１０３は、さらに残りのコードを利用したい場合には、コード認証機関１０２に対価を支払い（Ｐ４５）、残りのコードの提供を受ける（Ｐ４４）。

【００８０】また、図１９と同様に、コード作成者１０１がコード認証機関１０２に手数料を支払うことで、コードに検証用情報を付加してもらうことも可能である。この場合、コード作成者１０１がコード利用者１０３に

対して残りの部分のコードの対価を提示し、コード利用者１０３に対してコードを提供すると同時に対価を徴収する。

【００８１】図２２は、このようなコード配布方法を示している。図２２において、Ｐ５１、Ｐ５２、およびＰ５３の処理は、図１９のＰ２１、Ｐ２２、およびＰ２３の処理と同様である。次に、コード作成者１０１は、認証済みコードの一部を、例えば、上述したような方法で無償配布する（Ｐ５４）。これを入手して利用したコード利用者１０３が残りのコードを利用したい場合には、対価を支払い（Ｐ５６）、残りのコードを入手する（Ｐ５５）。

【００８２】また、コード作成者１０１の代わりに、コード認証機関１０２がコードを配布してもよい。この場合、コード認証機関１０２が、コード利用者１０３に対して残りの部分のコードの対価を提示し、コード利用者１０３に対してコードを提供すると同時に対価を徴収し、徴収した対価をコード作成者１０１に支払う。

【００８３】以上説明したようなコード配布方法によれば、コード認証機関による認証済みのコードが配布されるので、ユーザは、安心してコードを利用することができる。これにより、コードを利用するユーザが増加し、コードを広範囲に配布することが可能となる。

【００８４】ところで、図５のセキュアＯＳ５１は、例えば、セキュアドライブ／メディア２５にあらかじめ格納され、必要に応じてメモリにロードされて、動作を開始する。また、セキュアＯＳ５１を外部に保存しておき、必要に応じてシステムにインストールすることも可能である。

【００８５】図２３は、セキュアＯＳ５１を含むプログラムとデータをマルチプロセッサシステムに供給することのできるコンピュータ読み取り可能な記録媒体を示している。

【００８６】サーバ１１１のデータベース１１２や可搬記録媒体１１３に保存されたプログラムとデータは、マルチプロセッサシステムのメモリ１１４にロードされる。このとき、サーバ１１１は、プログラムとデータを搬送する搬送信号を生成し、ネットワーク上の任意の伝送媒体を介してマルチプロセッサシステムに送信する。そして、マルチプロセッサシステムは、そのデータを用いてそのプログラムを実行し、必要な処理を行う。

【００８７】可搬記録媒体１１３としては、メモリカード、フロッピー（登録商標）ディスク、ＣＤ－ＲＯＭ、光ディスク、光磁気ディスク等、任意のコンピュータ読み取り可能な記録媒体が用いられる。また、メモリ１１４は、図２の通常メモリ２１またはセキュアメモリ２３、図１２のセキュアメモリ９１、あるいは図１３の共有メモリ９３に対応する。

（付記１）　マルチプロセッサシステムを用いたコード実行装置であって、セキュアタスクの暗号化コードと、該暗号化コードが正当であることを検証するための検証用情報とを格納するセキュアメモリ手段と、前記検証用情報により前記暗号化コードが正当であると検証されたとき、該暗号化コードを実行するセキュアプロセッサ手段と、通常タスクのコードを格納する通常メモリ手段と、前記通常タスクのコードを実行する通常プロセッサ手段と、前記セキュアタスクと通常タスクの振り分けを行って、前記暗号化コードを前記セキュアメモリ手段に格納し、前記通常タスクのコードを前記通常メモリ手段に格納する制御手段とを備えることを特徴とするコード実行装置。

（付記２）　前記セキュアメモリ手段は、物理メモリ割り当ての単位毎に前記暗号化コードを格納し、該単位毎の暗号化コードに対する検証用情報を格納し、該検証用情報を用いて該単位毎の暗号化コードを検証し、前記セキュアプロセッサ手段は、正当であると検証された暗号化コードに含まれる暗号化命令をフェッチし、復号して実行することを特徴とする付記１記載のコード実行装置。

（付記３）　前記セキュアプロセッサ手段は、複数の復号鍵を保持し、該複数の復号鍵のうち指示された復号鍵を用いて、前記暗号化命令を復号することを特徴とする付記２記載のコード実行装置。

（付記４）　前記セキュアメモリ手段とセキュアプロセッサ手段は、相互認証を行ってセッション鍵を共有し、該セキュアメモリ手段は、前記暗号化命令を該セッション鍵でさらに暗号化して該セキュアプロセッサ手段に転送することを特徴とする付記２記載のコード実行装置。

（付記５）　前記暗号化コードを固有鍵でさらに暗号化して格納するセキュアドライブ手段をさらに備え、該セキュアドライブ手段と前記セキュアメモリ手段は、相互認証を行ってセッション鍵を共有し、該セキュアドライブ手段は、前記制御手段からの読み出し指示に基づいて、該暗号化コードを該固有鍵で復号し、該セッション鍵で暗号化して、該セキュアメモリ手段に転送することを特徴とする付記１記載のコード実行装置。

（付記６）　前記セキュアメモリ手段と通常メモリ手段の領域の少なくとも一部がオーバーラップしていることを特徴とする付記１記載のコード実行装置。

（付記７）　前記セキュアプロセッサ手段は、前記暗号化コードを用いて、該暗号化コードを実行する論理回路の少なくとも一部を、回路状態として不揮発的に固定することを特徴とする付記１記載のコード実行装置。

（付記８）　前記セキュアプロセッサ手段は、前記論理回路の前の回路状態を消去して、新しく上書きすることを特徴とする付記７記載のコード実行装置。

（付記９）　物理メモリ割り当ての単位毎に暗号化コードを格納する手段と、該単位毎の暗号化コードが正当であることを検証するための検証用情報を格納する手段と、該検証用情報を用いて該単位毎の暗号化コードを検

17

証する手段とを備えることを特徴とするメモリ。

（付記１０）　暗号化コードを格納するメモリから、該暗号化コードが正当であることを示す通知を受け取る手段と、前記通知を受け取ったとき、前記暗号化コードに含まれる暗号化命令をフェッチして復号する手段と、復号された命令を実行する手段とを備えることを特徴とするプロセッサ。

（付記１１）　コンピュータのためのプログラムを記録した記録媒体であって、該プログラムは、セキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含むマルチプロセッサシステムにおいて、該セキュアタスクと通常タスクの振り分けを行い、前記セキュアタスクの暗号化コードと、該暗号化コードが正当であることを検証するための検証用情報とを、セキュアメモリに格納し、前記検証用情報により前記暗号化コードが正当であると検証されたとき、該暗号化コードをセキュアプロセッサに実行させる処理を前記コンピュータに実行させることを特徴とするコンピュータ読み取り可能な記録媒体。

（付記１２）　セキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含むマルチプロセッサシステムにおいて、該セキュアタスクと通常タスクの振り分けを行い、前記セキュアタスクの暗号化コードと、該暗号化コードが正当であることを検証するための検証用情報とを、セキュアメモリに格納し、前記検証用情報により前記暗号化コードが正当であると検証されたとき、該暗号化コードをセキュアプロセッサに実行させる処理をコンピュータに実行させるためのプログラム。

（付記１３）　コンピュータのためのプログラムを該コンピュータに搬送する搬送信号であって、該プログラムは、セキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含むマルチプロセッサシステムにおいて、該セキュアタスクと通常タスクの振り分けを行い、前記セキュアタスクの暗号化コードと、該暗号化コードが正当であることを検証するための検証用情報とを、セキュアメモリに格納し、前記検証用情報により前記暗号化コードが正当であると検証されたとき、該暗号化コードをセキュアプロセッサに実行させる処理を前記コンピュータに実行させることを特徴とする搬送信号。

（付記１４）　コード作成者が、コード認証機関に実行可能なコードを提供し、前記コード認証機関が、前記コードが正当であることを検証するための検証用情報を該コードに付加して、マルチプロセッサシステムのユーザに配布し、前記マルチプロセッサシステムは、前記コードを用いてセキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含み、該セキュアタスクと通常タスクの振り分けを行い、前記検証用情報を用いて該コードが正当であることを検証し、

18

該コードを実行することを特徴とするコード配布方法。

（付記１５）　前記コード認証機関は、前記コード作成者に対して対価を提示して前記コードを収集し、収集に際して対価を支払い、前記ユーザに対して該コードの対価を提示し、前記検証用情報を付加した後、該ユーザに対して該コードを提供すると同時に対価を徴収することを特徴とする付記１４記載のコード配布方法。

（付記１６）　前記コード認証機関は、前記コードを２つ以上の部分に分割し、最初に一部を配布し、その後、前記ユーザの要求に応じて、残りの部分を配布することを特徴とする付記１４記載のコード配布方法。

（付記１７）　前記コード認証機関は、前記コード作成者に対して対価を提示してコードを収集し、収集に際して対価を支払い、前記ユーザに対して前記残りの部分の対価を提示し、検証用情報を付加した後、コードを提供して対価を受領することを特徴とする付記１６記載のコード配布方法。

（付記１８）　コード作成者が、コード認証機関に実行可能なコードを提供して、手数料を支払い、前記コード認証機関が、前記コードが正当であることを検証するための検証用情報を該コードに付加し、前記コード作成者が、前記コードをマルチプロセッサシステムのユーザに配布して、該ユーザが支払う対価を受領し、前記マルチプロセッサシステムは、前記コードを用いてセキュアタスクを実行するセキュアプロセッサと、通常タスクを実行する通常プロセッサを含み、該セキュアタスクと通常タスクの振り分けを行い、前記検証用情報を用いて該コードが正当であることを検証し、該コードを実行することを特徴とするコード配布方法。

（付記１９）　前記コード作成者は、前記コードを２つ以上の部分に分割し、最初に一部を配布し、その後、前記ユーザの要求に応じて、残りの部分の対価を提示し、コードを提供して対価を受領することを特徴とする付記１８記載のコード配布方法。

【００８８】

【発明の効果】本発明によれば、セキュアプロセッサを含むヘテロなマルチプロセッサシステムを構築して、セキュアタスクと非セキュアタスクを振り分けることで、ＯＳによる制御が簡単になり、セキュア処理を効率よく実行することができる。また、セキュアタスクのコードに署名を施す際に、メモリに読み込む部分毎に署名を付与することで、コードを効率よく実行することが可能となる。

【図面の簡単な説明】

【図１】本発明のコード実行装置の原理図である。

【図２】第１のマルチプロセッサシステムの構成図である。

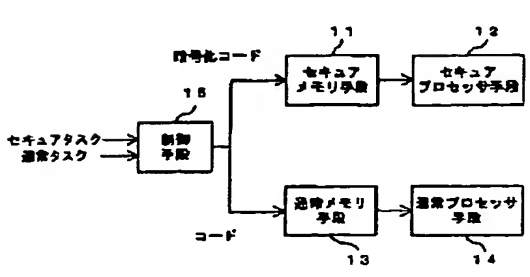
【図３】セキュアメモリとセキュアプロセッサの第１の構成図である。

【図４】セキュアプロセッサの構成図である。

【図５】セキュアＯＳの構成図である。
【図６】セキュアタスク管理の処理のフローチャートである。
【図７】セキュアメモリ管理の処理のフローチャートである。
【図８】実メモリ解放処理のフローチャートである。
【図９】セキュアファイルシステムの処理のフローチャートである。
【図１０】セキュアメモリとセキュアプロセッサの第２の構成図である。
【図１１】セキュアドライブ／メディアとセキュアメモリの構成図である。
【図１２】第２のマルチプロセッサシステムの構成図である。
【図１３】第３のマルチプロセッサシステムの構成図である。
【図１４】回路生成処理のフローチャートである。
【図１５】基本回路の配列を示す図である。
【図１６】演算器群を示す図である。
【図１７】第１のコード配布方法を示す図である。
【図１８】第１の対価の支払いを示す図である。
【図１９】第２のコード配布方法を示す図である。
【図２０】第３のコード配布方法を示す図である。
【図２１】第２の対価の支払いを示す図である。
【図２２】第４のコード配布方法を示す図である。
【図２３】記録媒体を示す図である。
【符号の説明】
１１　セキュアメモリ手段
１２　セキュアプロセッサ手段
１３　通常メモリ手段
１４　通常プロセッサ手段
１５　制御手段
２１　通常メモリ
２２　通常プロセッサ
２３、９１　セキュアメモリ
２４　セキュアプロセッサ
２５　セキュアドライブ／メディア
２６、９２、９４、９５、９６　システムバス
３１　認証局公開鍵
３２　署名検証部
３３　署名保持部
３４　ページ
４１　復号鍵設定部
４２　復号鍵保持部
４３、６３、７２、８４　復号部
４４　プロセッサ
４５　復号鍵指示部
５１　セキュアＯＳ
５２　セキュアタスク管理
５３　セキュアメモリ管理
５４　セキュアファイルシステム
６１、７１、８３　相互認証／セッション鍵共有部
６２、８５　暗号部
８１　格納媒体
８２　格納装置固有鍵
８６　格納媒体固有鍵
８７　暗号化コード
９３　共有メモリ
１０１　コード作成者
１０２　コード認証機関
１０３　コード利用者
１１１　サーバ
１１２　データベース
１１３　可搬記録媒体
１１４　メモリ

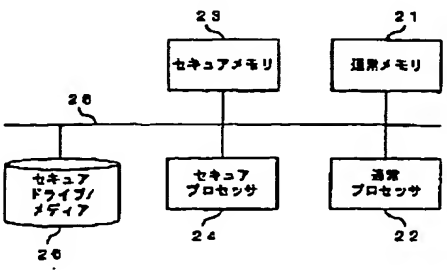
【図１】

本　発　明　の　原　理　図



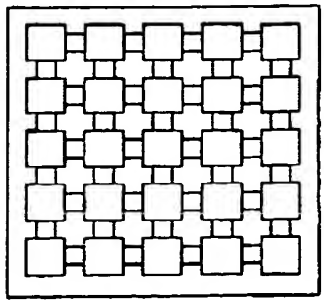
【図２】

第１のマルチプロセッサシステムの構成図



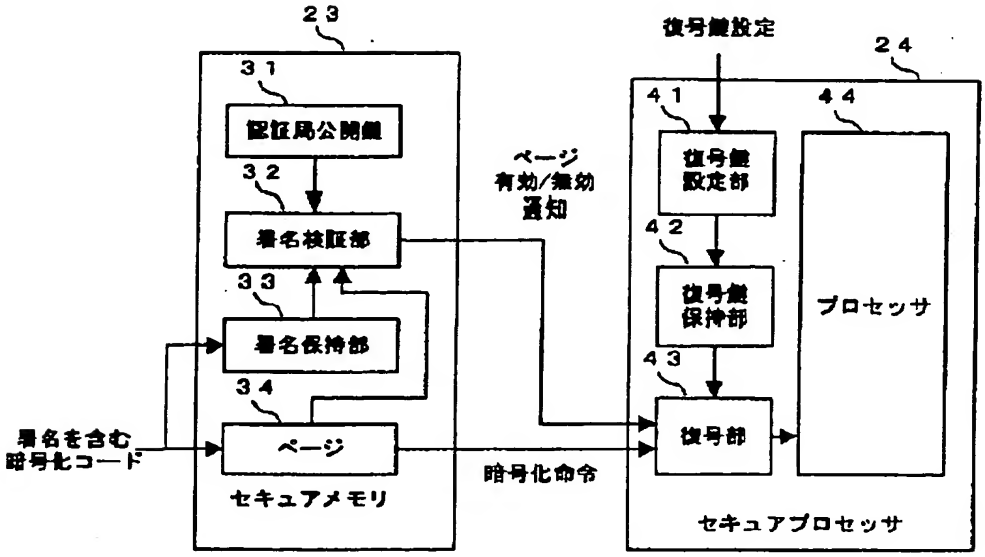
【図１５】

基本回路の配列を示す図



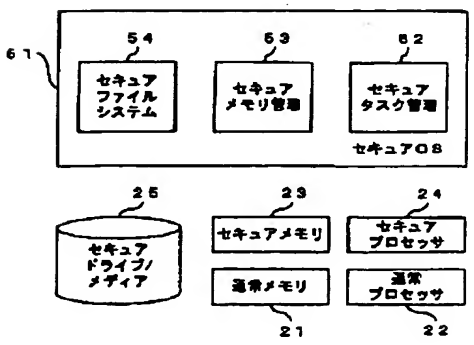
【図3】

セキュアメモリとセキュアプロセッサの第1の構成図



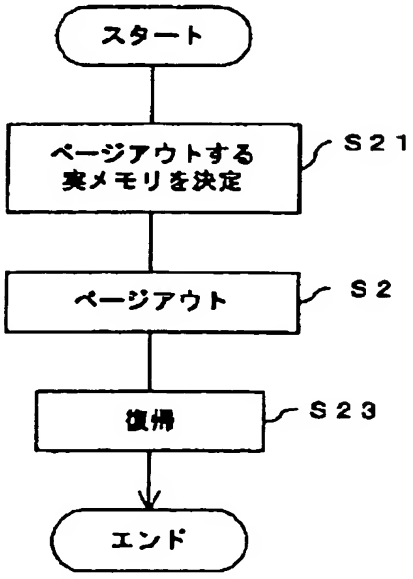
【図5】

セキュアOSの構成図



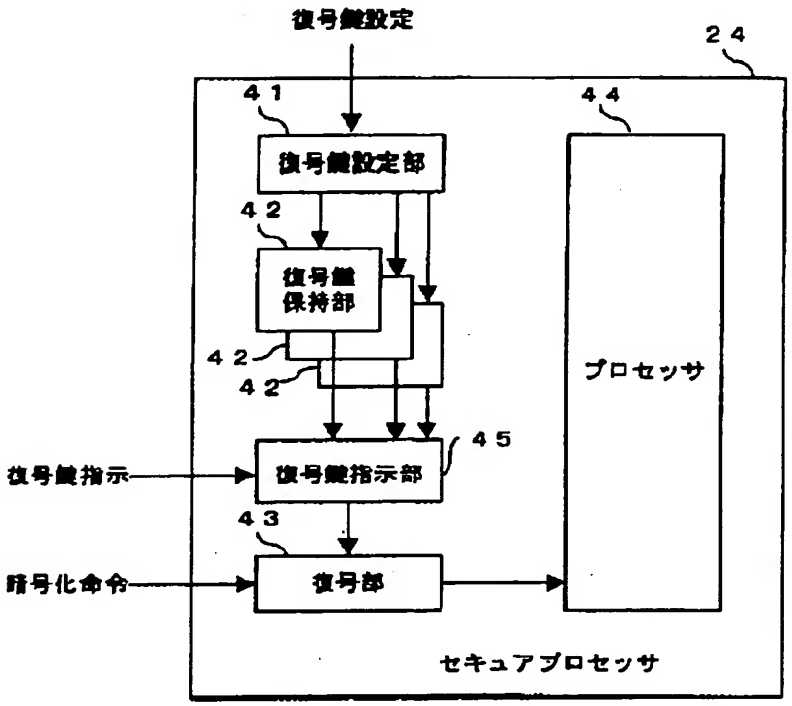
【図8】

実メモリ解放処理のフローチャート



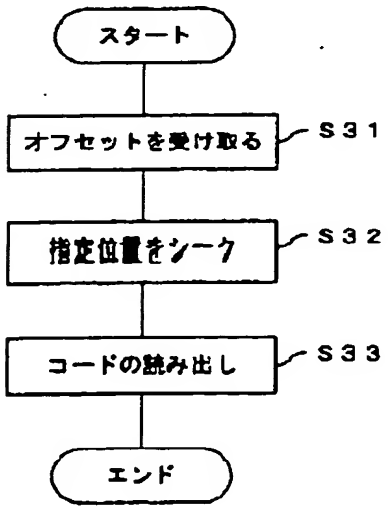
【図４】

セキュアプロセッサの構成図



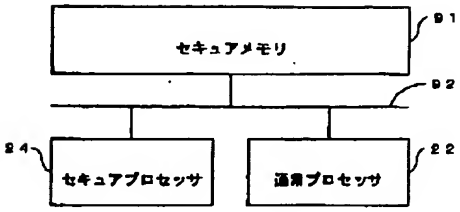
【図９】

セキュアファイルシステムの処理のフローチャート



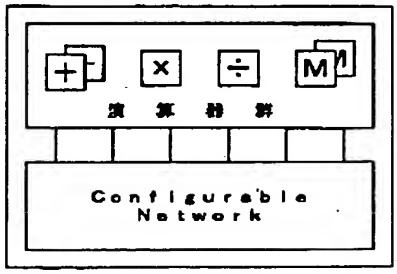
【図１２】

第２のマルチプロセッサシステムの構成図



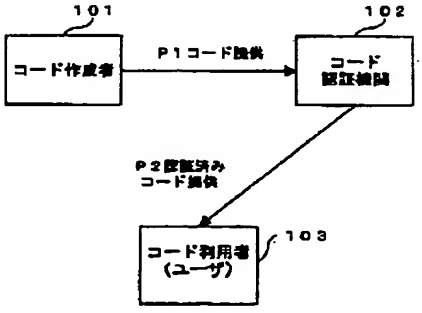
【図１６】

演算器群を示す図



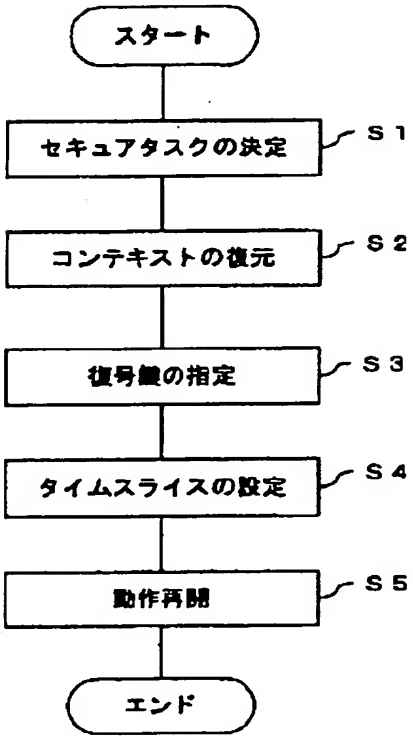
【図１７】

第１のコード配布方法を示す図



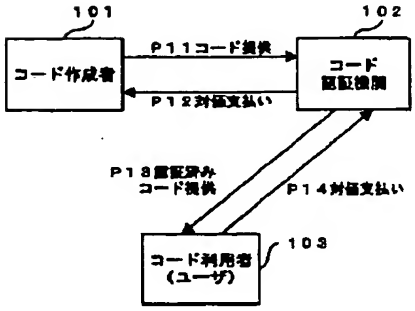
【図６】

セキュアタスク管理の処理へのフローチャート



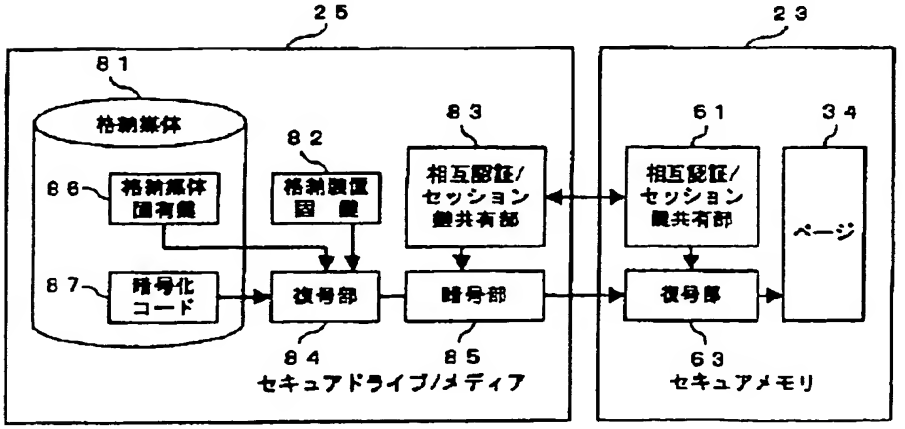
【図１８】

第１の対価の支払いを示す図



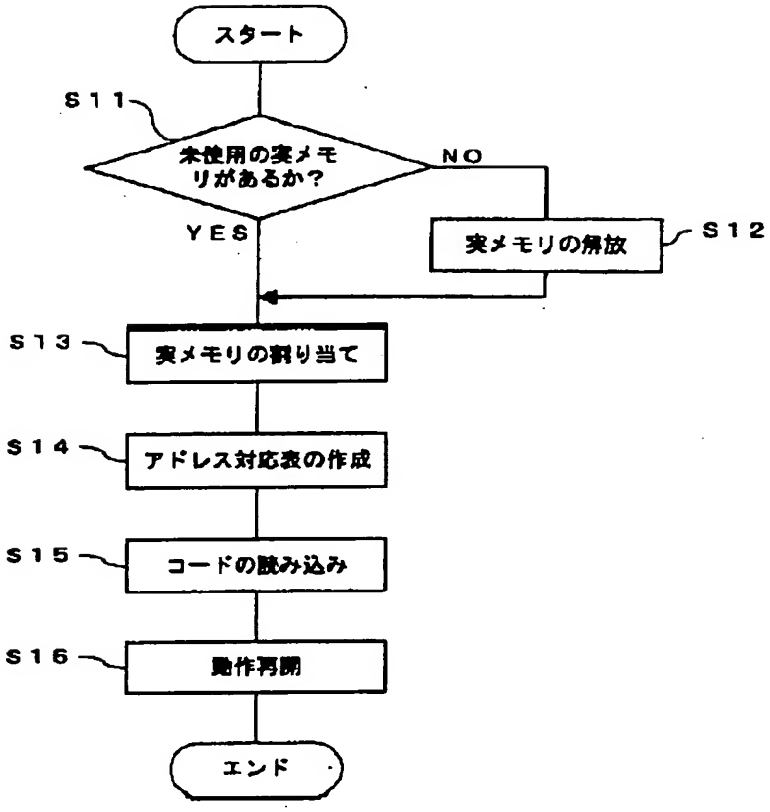
【図１１】

セキュアドライブ/メディアとセキュアメモリの構成図



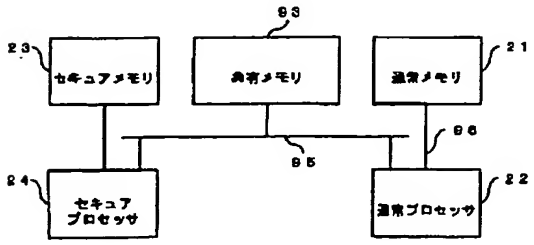
【図7】

セキュアメモリ管理の処理のフローチャート



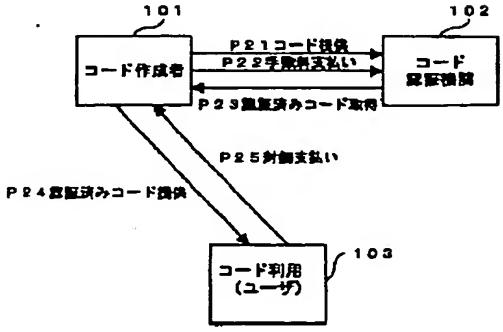
【図13】

第3のマルチプロセッサシステムの構成図



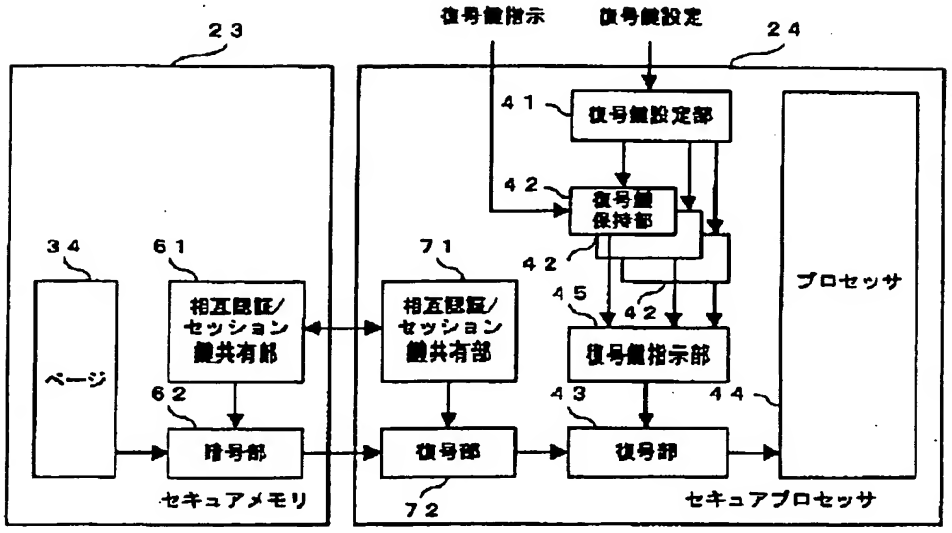
【図19】

第2のコード配布方法を示す図



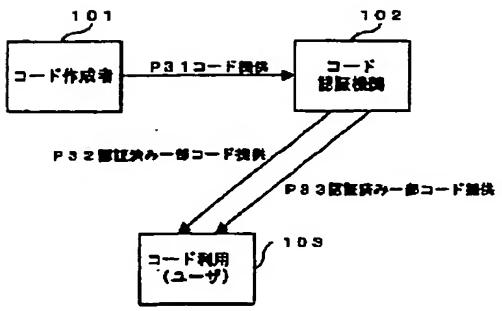
【図１０】

セキュアメモリとセキュアプロセッサの第２の構成図



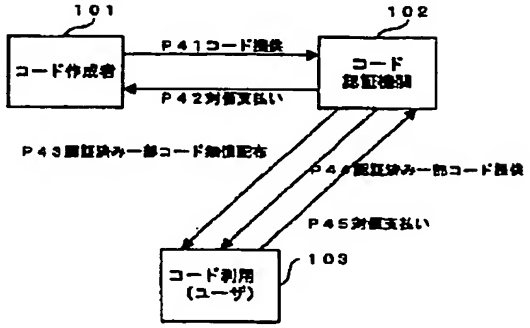
【図２０】

第３のコード配布方法を示す図



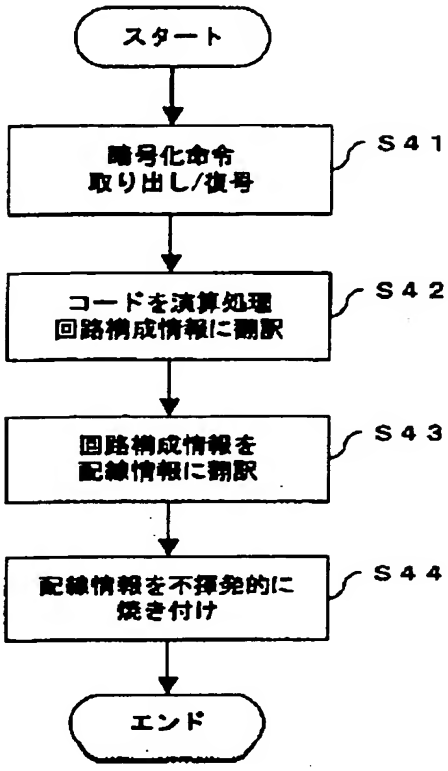
【図２１】

第２の対価の支払いを示す図



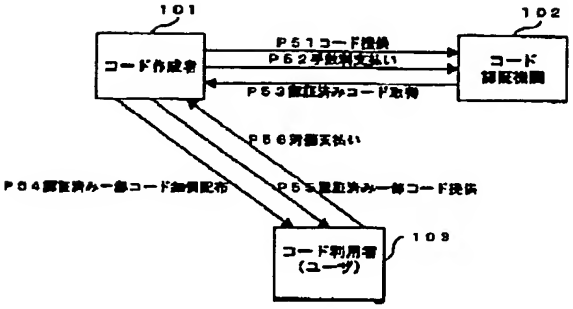
【図１４】

回路生成処理のフローチャート



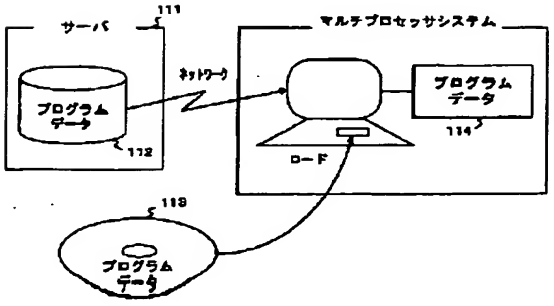
【図２２】

第４のコード配布方法を示す図



【図２３】

記　録　媒　体　を　示　す　図



フロントページの続き

(51)Int.Cl.[7]	識別記号	FI		テーマコード(参考)
G06F 15/16	610	G06F	9/06	660G 5J104
H04L 9/08			9/30	320C

Fターム(参考) 5B017 AA03 BA07 CA15
5B033 BB03
5B045 DD01 GG06 GG09
5B076 FA01 FD04
5B098 AA10 GA04 GC01
5J104 AA09 EA06 LA03 NA02

Code xecution apparatus and code distributing method

Patent Number: US2002184046
Publication date: 2002-12-05
Inv ntor(s): KAMADA JUN (JP); KOTANI SEIGO (JP)
Applicant(s): FUJITSU LTD (JP)
Requested Patent: JP2002353960
Application Number: US20020042262 20020111
Priority Number(s): JP20010162271 20010530
IPC Classification: G06F17/60
EC Classification:
Equivalents: EP1278114

Abstract

In a heterogeneous multiprocessor system having a secure processor and a normal processor, a secure task and an unsecured task are allocated to respective processors. An encrypted code of the secure task is stored in a secure memory, and the secure memory verifies a signature using a public key of a certificate authority, and notifies the secure processor of the validity of the encrypted code. The secure processor fetches an encrypted instruction from the secure memory, and decrypts and executes the instruction